Fast binary support vector machine learning method by samples reduction

Abdelhamid Djeffal* and Mohemed Chaouki Babahenini

Department of Computer Science,
LESIA Laboratory,
Biskra University,
BP 145, 07000 Biskra, Algeria
Email: abdelhamid_djeffal@yahoo.fr
Email: chaouki.babahenini@gmail.com
*Corresponding author

Abdelmalik Taleb-Ahmed

LAMIH Laboratory,
University of Valenciennes,
59313, Valenciennes, cedex 9, France
Email: abdelmalik.taleb-ahmed@univ-valenciennes.fr

Abstract: Support vector machine is a well-known method of statistical learning by its good accuracy; however, its training time is very poor especially in case of huge databases. Many research works aim to reduce training samples to improve training time without significant loss in accuracy. In this paper, we propose a method called CB-SR, based on filtering and revision stages to eliminate samples that have little influence on learning results. Filtering stage uses a covering-based principle of samples to eliminate those faraway from decision boundaries and keep the closest ones. Revision stage allows to add after the first learning, samples eventually discarded by mistake. The results we obtain show the benefits of our approach over others existing ones.

Keywords: support vector machine; binary SVM; samples reduction; fast training; support vectors; separating hyperplane; separating margin; decision boundaries; statistical learning; machine learning.


Biographical notes: Djeffal Abdelhamid received his PhD in Computer Science from the Biskra University of Algeria, Master in Image Processing and IA from the Biskra University in 2004, and Engineer degree in Computer Science from the National Institute for Computer Science at Algiers in 1997. He has been a Lecturer at Computer Science Department at the University of Biskra since December 2004, and has been a member of LESIA Laboratory and research team in image processing and satellite images since January 2005. His current interests include: data mining, image processing, and machine learning.
Babahenini Mohamed Chaouki is an Associate Professor at Computer Science Department of Biskra University, Algeria. He received his MS from the Setif University and PhD in Computer Science from the Biskra University in 2006. His current research focuses on computer graphics, computer vision, data mining, etc.

Abdelmalik Taleb-Ahmed received his PhD in 1992 in the University of Sciences and Technologies from Lille France. He is an Assistant Professor from 1992 to 2003 in the Calais University. Since 2004, he has been a Professor at the University of Valenciennes. He works in LAMIH Laboratory UMR CNRS UVHC 8530 in Valenciennes. His current scientific interests include: image segmentation, prior knowledge integration in image analysis, partial differential equations and variational methods in image analysis, medical image analysis, including multimodal image registration, segmentation, computer-assisted surgery, diffusion MRI, etc.

1 Introduction

Among statistical learning methods, support vector machine (SVM) has very successful applications in several areas such as image recognition (face, fingerprint, …), finance (credit scoring, marketing, …), etc. Despite its excellent results in terms of accuracy and generalisation ability (Wang, 2005; Huang et al., 2006), the method suffers from the problem of learning time that is in the best case $O(nm^2)$ if $n$ is the number of training samples and $m$ is the number of attributes of each sample (Huang et al., 2006; Soman et al., 2009; Wang et al., 2012; Mahmoudreza and Hamidreza, 2014; Lin and Lin, 2003).

Since its proposal by Vladimir Vapnik in Vapnik (1995), the optimisation of the SVM method’s training time attracted the attention of many researchers. Some methods try to optimise the SVM implementation by using heuristics such as in Osuna et al. (1997). Other methods aim to reduce training samples and try to keep for training, only samples that might influence the generalisation capacity of the learned machine, i.e., sample candidates to be support vectors.

In literature, most proposed methods use clustering techniques to detect non-relevant samples for training. A preprocessing phase tries to build clusters for each class and then rejects samples that are near to clusters centres. Related work section exposes some of those methods.

In this paper, we propose a new method called covering-based samples reduction (CB-SR) that consists of two stages: a filtering stage and a revision stage. In filtering stage, instead of a classical clustering, we propose to verify, in the feature space, the coverage of a sample of a class against all the samples of the other class in binary SVM. If this sample is covered for all samples of the other class then it is removed from the training set otherwise it is kept. After this stage, a hyperplane separating the kept samples of the two classes is learned. In the revision stage, we classify all samples discarded in the filtering stage with the learned hyperplane. The samples that violate Karush-Kuhn-Tucker (KKT) conditions (Soman et al., 2009) are added to the filtered training set and a new learning phase is performed in which we start from the previous hyperplane in order to readjust it according to the added samples. The filtering phase significantly reduces the number of training samples by introducing the coverage principle, especially in case of datasets where the majority of the data do not lie at the
Fast binary support vector machine learning method by samples reduction

The revision stage allows to correct the situation of some samples that could be removed by errors in the filtering stage due to a possible mismatch between the chosen kernel and the problem to solve. Using several benchmark datasets, we brought advantages in terms of learning speed and recognition rate, brought by our solution. The rest of the paper is organised as follows: In Section 2, the principle of the SVM method is introduced and in Section 4, a detailed description of the proposed method is presented. Section 5 shows experimental results. The paper ends with a conclusion in Section 6.

2 Binary SVM principle (Wang, 2005)

Consider the problem of separating two classes represented by \( n \) samples:
\[
\{(x_i, y_i), \ldots, (x_n, y_n)\}, \quad x_i \in \mathbb{R}, \quad y_i \in \{-1, +1\}
\]
where \( x_i \) are learning samples and \( y_i \) their respective classes. The objective of the SVM method is to find a linear function \( f \) [equation (1)], called hyperplan, that can separate the two classes:
\[
\begin{cases}
  f(x) = (x \cdot w) + b; \\
  f(x) > 0 \Rightarrow x \in \text{class} +1 \\
  f(x) < 0 \Rightarrow x \in \text{class} -1
\end{cases}
\]

where \( x \) is a sample to classify, \( w \) is a vector and \( b \) is a bias. We must therefore find the widest margin between two classes, which means minimising \( \frac{1}{2} ||w||^2 \). In cases where training data are not linearly separable, we allow errors \( \xi \) (called slack variables) of samples from boundaries of the separation margin with a penalisation parameter \( C \) and so the problem becomes a convex quadratic programming problem:

\[
\begin{aligned}
\text{Minimise} & \quad \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \xi_i \\
\text{under constraints} & \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1..n \\
& \quad \xi_i \geq 0
\end{aligned}
\]

The problem of equation (2) can be solved by introducing Lagrange multipliers in the following dual problem:

\[
\begin{aligned}
\text{Minimise} & \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i \\
\text{under constraints} & \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& \quad 0 \leq \alpha_i \leq C
\end{aligned}
\]
Hence, we can have the following decision function (hyperplane):

\[
H(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b
\]

The function \( K \) is called \textit{kernel}, it is a symmetric function that satisfies Mercer conditions (Liu and Feng, 2008). It can represent a transformation of the original input space in which data could be nonlinearly separable to a new larger space where a linear separator exists. Solving equation (3) means to find \( \alpha_i \) values. Samples, with corresponding Lagrange multipliers not null (\( \alpha_i \neq 0 \)), are called support vectors and they represent samples falling on separation boundaries between the two classes. According to Vapnik and Vapnik (1998), support vectors’ number is very less to the number of all training samples. For examples, in image processing field, they represent only 10% of training samples.

Solving the problem of equation (3) requires an optimisation especially when the number of samples is high. Among the optimisation methods, most commonly used, there is the sequential minimal optimisation (SMO) where the problem is divided into several sub-problems, each optimises two \( \alpha_i \) (Osuna et al., 1997).

3 Related work

In Lin and Lin (2003), authors start by performing a K-means clustering to detect clusters. Then they detect crisp clusters representing clusters with samples belonging to the same class. The last step eliminates samples out of a defined safety region.

In Liu and Feng (2008), authors propose a hierarchical clustering method that only keeps reduced datasets that represent all the training data. Obtained results improve, indeed, the learning speed with sometimes insignificant loss in accuracy, but clustering duration is considerable.

In Comak and Arslan (2008), in addition to a simple clustering, authors proposed a non-candidate samples elimination based on a K-nearest neighbours (K-NN), but it suffers from the same problem: clustering duration.

In Song et al. (2008), authors proposed a method based on K-closest sub-clusters. The start by clustering the two classes and then take only samples of k closest clusters of each class to the other.

In Lin and Yeh (2009), authors propose a reduction method based on genetic algorithms with elitism. They start from a population of randomly chosen solutions, and use an appropriate fitness function to achieve the optimal set of support vectors.

A fuzzy c-means clustering-based method was proposed in Wang et al. (2012). It uses FCM algorithm to detect clusters in both positive and negative classes. For each cluster, it uses multivariate Gaussian distribution (MGD) to reject samples near to clusters’ centres.

In Lu et al. (2010), authors perform two steps after clustering to reduce training samples. The first step rejects outlier samples and the second rejects samples situated behind clusters’ centres.
4 Proposed method description

In our proposal and instead of using clustering techniques, we use a covering principle to discover samples to be rejected. This principle enormously reduces training time compared to clustering techniques.

4.1 Flowchart of the method

The scheme of our method is illustrated in the flowchart of Figure 1.

Our solution is based mainly on the following filtering and revision stages.

Figure 1 Flowchart of the proposed method

4.2 Filtering

The used filtering reduces the number of training samples in order to accelerate training speed. It is based on the concept of covering samples of a class against those of the opposite class. A sample covered by samples of its class against all the samples of the other class is a sample that has a low probability to become a support vector. Coverage of a sample can be defined as follows:
Definition 1: The coverage area of a sample \( x_i \) of a class \( y_i \) against a sample \( x_k \) of the class \( y_k = -y_i \), is defined by the area bounded by two hyperplanes \( H_{ik}^+ \) and \( H_{ik}^- \) with maximum margin, passing by the two samples \( x_i \) and \( x_k \). This area can be controlled by the parameter \( \rho \) to adjust it depending on the degree of desired filtering (cf., Figure 2).

Figure 2  Coverage of a sample by another of the same class

Indeed, a sample \( x_i \) of the class \( y_i (\pm 1) \) is covered by another sample \( x_j \) of the same class against a sample \( x_k \) of the class \( -y_i \), if \( x_j \) lies in coverage area of \( x_i \). In Figure 2, \( x_i \) is outside the coverage area of \( x_j \) bounded by \( H_{ik}^+ \) and \( H_{ik}^- \), while \( x_j \) lies in the coverage area, of \( x_i \) against \( x_k \), bounded by \( H_{ik}^+ \) and \( H_{ik}^- \).

More formally, the idea of SVMs is to find a hyperplane that maximises the margin of separation between two classes. The problem is formulated by the following dual problem [equation (5)]:

\[
\text{maximize} \quad \frac{1}{2} \langle W, W \rangle - C \sum_{i=1}^{n} \alpha_i \left( 1 - y_i \langle x_i, W \rangle \right)
\]

\[
\text{subject to} \quad 0 \leq \alpha_i \leq C
\]
Fast binary support vector machine learning method by samples reduction

Maximise \( Q(\alpha) = \sum_{i=1}^{n} \alpha_i \)

\[
\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) 
\]

under constraints

\[
\sum_{i=1}^{n} \alpha_i y_i = 0 \\
0 \leq \alpha_i \leq C
\]

where \( x_i \) are training samples, \( y_i \) their respective classes (±1), \( \alpha_i \) are the Lagrange multipliers, \( C \) is a penalisation parameter on \( \alpha_i \) and \( K \) is a kernel. Once the optimum \( \alpha_i \) determined, each new sample can be classified by the following function \( H \):

\[
H(x) = \sum \alpha_i y_i K(x_i, x) + b
\]

with

\[
b = \frac{1}{n} \sum_{i=1}^{n} y_i - \frac{1}{n} \sum_{j=1}^{n} \alpha_i y_j K(x_j, x_i)
\]

Let us return to our example in Figure 2, and calculate the separation function \( H_{ik} \) allowing the separation between the two samples \( x_i \) and \( x_k \). We have only two samples \( x_i \), \( x_k \) of different classes, hence, \( \alpha_i = \alpha_k = [\) the first constraint of the equation (5)] and \( Q(\alpha) \) becomes:

\[
Q(\alpha) = 2\alpha - \frac{1}{2} D \alpha^2 = 2\alpha - \frac{D}{2} \alpha^2
\]

with

\[
D = K(x_i, x_i) - 2K(x_i, x_k) + K(x_k, x_k)
\]

And as we seek to maximise \( Q \), we must find \( \alpha \) that annuls the derivative of \( Q \), i.e., \( \alpha = \frac{2}{D} \). The separating function \( H_{ik} \) between \( x_i \) and \( x_k \) becomes:

\[
H_{ik}(x) = \frac{1}{D} (K(x_i, x) - K(x_k, x)) + \frac{\alpha}{2} - \frac{D}{2} \alpha^2
\]

with

\[
b = \frac{K(x_i, x_i) - K(x_k, x_k)}{D}
\]

And there, we can verify coverage of \( x_j \) to \( x_i \) against \( x_k \). We say that \( x_j \) covers \( x_i \) against \( x_k \) if:

\[
\rho - 1 \leq H_{ik}(x_i) \leq 1 - \rho
\]
The parameter $\rho$ can control the coverage margin of $x_i$, more it is high, less is the coverage margin, and less is the mutual coverage probability (Figure 3) and is fortunate to $x_i$ to be discarded from the training samples. This parameter will allow us to control the level of desired filtering.

One can quickly notice that the condition of equation (9) is not sufficient to discard $x_i$ when learning, since the two samples $x_i$ and $x_j$ can cover each other, i.e., can both be candidates (cf., Figure 3).

**Figure 3** Mutual coverage

To discard $x_i$ vis-a-vis $x_j$, we must have $x_j$ covers $x_i$ and $x_i$ does not cover $x_j$ (case of Figure 2). From this definition of the coverage principle, we can define the total coverage of a sample:

**Definition 2:** We say that a sample $x_i$ of a class $y_i (\pm 1)$ is totally covered against the class $-y_i$ if for each sample $x_k$ of the class $-y_i$, there exists a sample $x_j$ of the class $y_i$ that covers $x_i$ against $x_k$. In other words:

$$x_i \in y_i \text{ is discarded if } \forall x_k \in -y_i, \exists x_j \in y_i / x_j \text{ covers } x_i \text{ and } x_i \text{ does not cover } x_j.$$  

This rule is used to filter the training dataset and exclude all samples totally covered. Algorithm 1 illustrates the followed steps.

It should be noted that filtering uses the same learning kernel, i.e., it does not depend on a specific kernel. We can see on the filtering algorithm that the number of iterations of “for” loops depends only on the number of samples of a single class. In addition, the size of the two classes decreases with the evolution of the algorithm since samples are rejected in each iteration.
Algorithm 1  Filtering

Input:  
- A set of training samples $S = \{(x_i, y_i), i = 1..n / x_i \in \mathbb{R}^n, y_i = \pm 1\}$,
- A kernel $K$ with its parameters,
- Filtering rate $\rho$.

Output:  
- A set of samples $S' = \{(x_i, y_i), i = 1..p, x_i \in \mathbb{R}, y_i = \pm 1 \text{ with } p \leq n\}$

Begin

$N^+ \leftarrow \{x_i / y_i = +1\}$ // positive samples
$N^- \leftarrow \{x_i / y_i = -1\}$ // negative samples

For all $x_i \in N^+$  
If Positive_Covered$(x_i)$ Then $N^+ \leftarrow N^+ - x_i$
EndFor

For all $x_i \in N^-$  
If Negative_Covered$(x_i)$ Then $N^- \leftarrow N^- - x_i$
EndFor

Return $S' = N^+ \cup N^-$

End.

Function Positive_Covered$(x_i)$ : boolean

Begin

For all samples $x_k \in N^-$
For all samples $x_j \in N^+$
If $[(\rho - 1 > H_{ik}(x_j) > 1 - \rho) \text{ or } (\rho - 1 \leq H_{jk}(x_i)) \text{ and } H_{jk}(x_i) \leq 1 - \rho]$ Then Return false
EndFor
EndFor

Return true

End;

Function Negative_Covered$(x_i)$ : boolean

Begin

For all samples $x_k \in N^+$
For all samples $x_j \in N^-$
If $[(\rho - 1 > H_{id}(x_i) > 1 - \rho) \text{ or } (\rho - 1 \leq H_{jd}(x_i)) \text{ and } H_{jd}(x_i) \leq 1 - \rho]$ Then Return false
EndFor
EndFor

Return true

End;

The algorithm is tested on a dataset of ten samples of positive class and ten samples of negative class. It allows, indeed, to keep only support vectors. Figure 4 illustrates this test where crossed samples represent discarded ones.

The set of filtered samples is passed then to the stage of 1st SVM learning to generate a hyperplane $H$ separating the two classes (Figure 1). This filtering is a complexity lower than that of the SMO learning algorithm. Indeed, the loops depend on the number of
positive or negative samples, not all samples. In addition, each discarded sample no longer participates in the rest of the algorithm which further reduces the duration of the filtering. The results will show later that the filtering time added to the learning time from the remaining samples is always very lower than the overall learning time from all samples.

**Figure 4** Filtering example

4.3 Revision

In this stage, the samples discarded in the previous stage, are verified by the obtained hyperplane. And since we assume that these samples are well classified, we put their $\alpha_i = 0$. We will therefore classify these samples by the hyperplane and check their status by the KKT conditions (Soman et al., 2009). KKT conditions are the conditions that must satisfy all $\alpha_i$ to stop the SVM training, they are given by 10:

$$
\begin{align*}
\alpha_i = 0 & \Rightarrow y_i f(x_i) > 1 \\
\alpha_i = C & \Rightarrow y_i f(x_i) \leq 1 \\
0 < \alpha_i < C & \Rightarrow y_i f(x_i) = 1
\end{align*}
$$

(10)

The examples that violate the KKT conditions are divided into three categories:

1. samples that fall in the separation margin and so they are considered well classified
2. samples that fall in the separation margin and so they are considered misclassified
3. samples that fall in the wrong side outside the separation margin but considered well classified.
Based on these conditions, we will classify the samples discarded in the filtering stage by the obtained hyperplane and add samples that violate the KKT conditions to the filtered set $S'$ to obtain a new set $S''$. $S''$ contains therefore the samples candidates to be support vectors and the samples misclassified by the hyperplane obtained after the filtering stage. A second SVM learning is performed to adjust our initial hyperplane that we do not change its $\alpha$ obtained in the first learning, with $\alpha$ of the added samples initialised to 0. Indeed, the chosen kernel may not have the needed skills to separate the two classes and can therefore give misclassified samples. In this revision stage, these samples are detected and the hyperplane is adjusted according to their positions.

5 Experimental results

5.1 Used datasets

Our method was first tested on toy (two dimensions) examples that we have chosen of different complexities. Then, we tested it on benchmark databases of binary classification on the site ‘machine learning repository UCI’ (Lin and Yeh, 2009). The used databases are shown in Table 1.

$n$ is the total number of records in the database, $N_a$ is the number of attributes, $N_t$ is the number of records used for training and $N_r$ is the number of records used for testing. German and diabetes databases were used to test the method KBK-SR (Liu and Feng, 2008) while tow spirals and tow ellipses databases were used to test the method of Lin and Yeh (2009).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$N_a$</th>
<th>$n$</th>
<th>$N_t$</th>
<th>$N_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germen</td>
<td>24</td>
<td>1,000</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>768</td>
<td>384</td>
<td>384</td>
</tr>
<tr>
<td>Tow spiras</td>
<td>2</td>
<td>3,000</td>
<td>300</td>
<td>2,700</td>
</tr>
<tr>
<td>Tow ellipses</td>
<td>2</td>
<td>3,000</td>
<td>300</td>
<td>2,700</td>
</tr>
<tr>
<td>Cancer analysis</td>
<td>32</td>
<td>596</td>
<td>596</td>
<td>596</td>
</tr>
<tr>
<td>Segmentation</td>
<td>19</td>
<td>2,309</td>
<td>2,309</td>
<td>2,309</td>
</tr>
</tbody>
</table>

5.2 Implementation and evaluation criteria

We implement our proposal, in Delphi programming language, where we implement SMO optimisation algorithm proposed in Osuna et al. (1997) and our two reduction stages for both graphic tests and databases tests.

The proposed method was tested on a Dual-core 1.6 GHZ with 1 GO memory. The used kernel is an RBF with a tube width $\sigma$ chosen empirically to start from the same recognition rates achieved in the references of comparison. It should be noted that we used, during the filtering, a tube width $\sigma = 3$. It is important to note also that the proposed filtering uses the same kernel of learning and the RBF kernel used here is chosen only to compare the results to those obtained in Liu and Feng (2009) and Lin and Yeh (2009). The evaluating of the performances of our methods is based on $n$ the number of samples.
kept after filtering, $T(s)$ the learning time in seconds based on $n$ samples and $R$ the recognition rate. We compared our method to two existing methods: KBK-SR (kernel bisecting k-means clustering) and Lin methods. The following section summarises the obtained results.

5.3 Results

The first tests have been performed on toys of different complexities and showed a great ability to discover the most important support vectors before proceeding to training. In the example of Table 2, we took 1,169 samples from two classes interfered in spiral. Classic SVM trained on all 1,169 samples gave a recognition rate of 100% in 432.656 seconds. Filtering stage has reduced training samples to only 471 and SVM training on these samples gave a recognition rate of 99.65 in 60.516 seconds. In the revision stage, 208 examples were added to recover the lost accuracy, which has cost an increase in the total training time (Filtering + 1st SVM + Revision + 2nd SVM) to 93.268 seconds.

The method was also tested on databases of reference: diabetes and German site ‘UCI machine learning repository’ (Asuncion and Newman, 2010) used in Liu and Feng (2008) to test the method KBK-SR. Table 3 summarises the results compared to those of Liu and Feng (2008) and Lin and Yeh (2009), where $n$ is the number of training samples, $\rho$ is the parameter controlling the filtering degree, $T$ is the training time in seconds and $R$ is the recognition rate.

Using all the 500 training samples of the German database, the classic SVM method gave a recognition rate of 72.8%. KBK-SR method (Liu and Feng, 2008) gave only 70.83% with 156 samples. Our method CB-SR was able to give a recognition rate of 72.8% only with three samples and the training time was reduced from 501.375 seconds to 0.578 seconds, which represent a considerable improvement. For the diabetes database, our method was able to achieve 70.31% with only ten samples. We explain this rate by the fact that the majority of this database samples are concentrated near the separation zone. The method was also tested on known databases in industrial simulation, used in Lin’s method (Lin and Yeh, 2009), our filtering has reduced the number of samples of tow ellipses database from 300 samples to only 12, while keeping the same accuracy (Table 3). The improvement is remarkable also in the tow spirals database, we achieved a recognition rate of 100% with only 37 samples.

The parameter $\rho$, used in our method, allows to control the filtering rate. A high value discards samples even if they are very close to the separation boundaries, which reduces the recognition rate. On the other hand, a low value of $\rho$ keeps samples though they are far from separation boundaries. Curves of Figure 5 show the filtering and recognition rates according to $\rho$ for German and tow ellipses databases. We can see that the parameter $\rho$ gives always good results between 0.6 and 0.99.

For automatic determination of $\rho$ parameter, we propose to initialise it to 1 then decrement it by a given step (e.g., 0.05) while observing the recognition rate. If it improves, the decrement is continued, otherwise it stops and we take the model of the iteration with best recognition rate (Figure 6).
Figure 5  Parameter $\rho$ influence on filtering and recognition rates (see online version for colours)

Table 2  Results on a toy example (see online version for colours)

<table>
<thead>
<tr>
<th></th>
<th>SVM</th>
<th>After filtering</th>
<th>After filtering and revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$T(s)$</td>
<td>$R$ (%)</td>
<td>$N$</td>
</tr>
<tr>
<td>1,169</td>
<td>432.656</td>
<td>100</td>
<td>471</td>
</tr>
</tbody>
</table>
### Table 3  
Results on different databases

<table>
<thead>
<tr>
<th>Database</th>
<th>Method</th>
<th>$N_f$</th>
<th>%</th>
<th>$\rho$</th>
<th>$T(s)$</th>
<th>R (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>SVM</td>
<td>500</td>
<td>100</td>
<td>/</td>
<td>501.375</td>
<td>72.8</td>
</tr>
<tr>
<td></td>
<td>KBK-SR</td>
<td>156</td>
<td>31.2</td>
<td>/</td>
<td>60.42</td>
<td>70.83</td>
</tr>
<tr>
<td></td>
<td>CB-SR</td>
<td>3</td>
<td>0.66</td>
<td>0.06</td>
<td>0.578</td>
<td>72.8</td>
</tr>
<tr>
<td>Diabetes</td>
<td>SVM</td>
<td>384</td>
<td>100</td>
<td>/</td>
<td>312.11</td>
<td>74.69</td>
</tr>
<tr>
<td></td>
<td>KBK-SR</td>
<td>120</td>
<td>31.25</td>
<td>/</td>
<td>28.25</td>
<td>73.54</td>
</tr>
<tr>
<td></td>
<td>CB-SR</td>
<td>10</td>
<td>2.6</td>
<td>0.22</td>
<td>0.001</td>
<td>70.31</td>
</tr>
<tr>
<td>Tow spirals</td>
<td>SVM</td>
<td>300</td>
<td>100</td>
<td>/</td>
<td>10.687</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Lin</td>
<td>20</td>
<td>6.66</td>
<td>/</td>
<td>/</td>
<td>90.35</td>
</tr>
<tr>
<td></td>
<td>CB-SR</td>
<td>20</td>
<td>6.66</td>
<td>0.01</td>
<td>0.380</td>
<td>99.18</td>
</tr>
<tr>
<td>Tow ellipses</td>
<td>SVM</td>
<td>300</td>
<td>100</td>
<td>/</td>
<td>12.395</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Lin</td>
<td>30</td>
<td>10</td>
<td>/</td>
<td>/</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>CB-SR</td>
<td>12</td>
<td>4</td>
<td>0.21</td>
<td>0.126</td>
<td>100</td>
</tr>
<tr>
<td>Cancer analysis</td>
<td>SVM</td>
<td>569</td>
<td>100</td>
<td>/</td>
<td>285.769</td>
<td>99.29</td>
</tr>
<tr>
<td></td>
<td>CB-SR</td>
<td>158</td>
<td>27.76</td>
<td>0.2</td>
<td>17.337</td>
<td>97.89</td>
</tr>
<tr>
<td>Segmentation</td>
<td>SVM</td>
<td>2309</td>
<td>100</td>
<td>/</td>
<td>84.61</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>CB-SR</td>
<td>32</td>
<td>1.38</td>
<td>0.6</td>
<td>0.937</td>
<td>100</td>
</tr>
</tbody>
</table>

**Figure 6** Automatic determination of $\rho$

![Diagram](image_url)
The application of this technique on German database allowed to achieve the highest recognition rate after the third iteration, which increased the training time of the method to 1.734 seconds but remains negligible compared to the training time of SVM method without filtering which was 501 seconds.

The revision stage is performed if the recognition rate after the first learning is not unsatisfactory. It allows to return to the recognition rate given by learning using all the examples, at minimum cost. Indeed, we only need to slightly modify the first hyperplane to return to the general case. The overall time learning after revision is still lower than using all the learning samples.

6 Conclusions

In this paper, we presented a new method for the discovery of samples candidates to be support vectors for SVM method, before starting training. The aim of our approach is to reduce the training time especially in the case of huge databases. The proposed method called CB-SR is divided into two phases filtering and revision. The filtering stage is based on the discarding of samples having a total coverage against samples of the opposite class and the revision stage allows to add samples discarded by mistake. We tested our method on several databases of reference and we have compared its results to other similar works. The obtained results are very convincing and very interesting in terms of learning time and recognition rate. We found that training time was reduced very significantly while maintaining accuracy. In our future work, we aim to study the use of coverage principle in one-class an multi-class SVM problems.

References


